

本周周报（6.1-6.7）

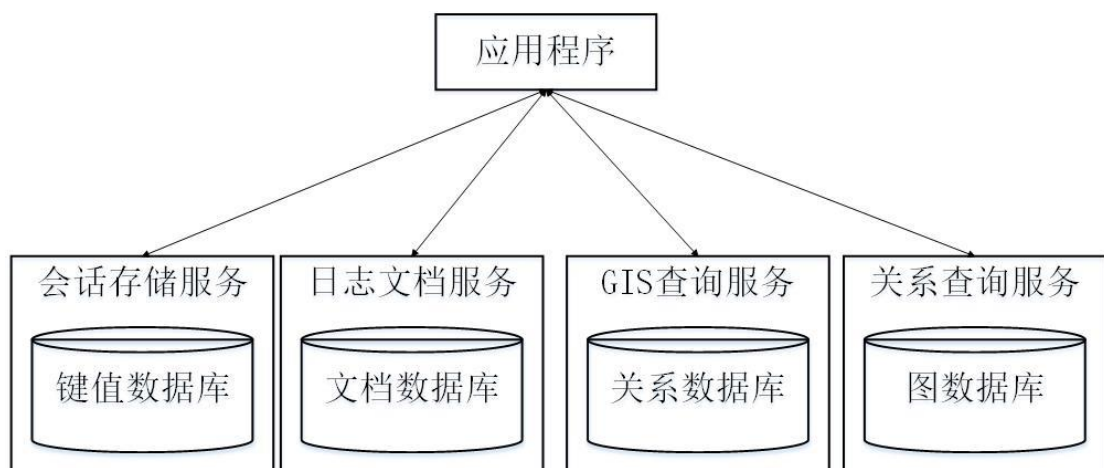
刘昊南

本周工作

1. 本周使用大的数据集(5000 万条)对 MongoDB 和 PostgreSQL 做详尽的插入、建索引和查询测试，老的硬盘在第一次测试时扛不住压力坏掉了，还损坏了同一块盘里的软件和文档，后换了一块新盘单独给数据库使用，并重新做了测试
2. 测试结果如下（MongoDB 和 PostgreSQL 各插入 5000 万条数据）：
 - a) 插入：PostgreSQL 花费 1 小时 6 分钟，MongoDB 只花费了 14 分钟，在插入速度上 MongoDB 完胜
 - b) 建索引：PostgreSQL 花费 33 分钟，MongoDB 花费 11 分钟，MongoDB 仍然要快许多
 - c) 磁盘空间：PostgreSQL 使用了 13GB，MongoDB 使用了 11GB，MongoDB 要略微好一点
 - d) 查询(120.83, 27.93)附近的 1000 个点：PostgreSQL 和 MongoDB 都只用了 3 秒钟
3. 分析：
 - a) 在插入、建索引、磁盘空间和查询几个方面 MongoDB 相对于 PostgreSQL 都不落下风，在插入和建索引的速度上还有巨大的优势
 - b) PostgreSQL 基于的 GIS 查询函数的丰富程度上有巨大的优势，MongoDB 只支持很少的几个基本查询操作符，比如 PostgreSQL 可以把查询出来的点聚合成一条轨迹以及计算轨迹长度等函数，但是 MongoDB 不支持
 - c) 跟斐然师兄讨论后，决定采用 PostgreSQL 存储 GIS 数据，理由如下：
 - i. 数据库的插入和建索引的速度对于我们来说不是那么重要，因为数据的大量集中插入只需要做一次，如果有实时数据进来那么每次插入的数据量不会很大，同时通过建立数据库集群把负载均衡到不同的数据库服务器可以极大改善插入速度
 - ii. 我们的数据库中每条记录都是一个点，并没有组织成轨迹存入数据库，否则会丢失时间信息且不能选择任意时间段的轨迹。使用 PostgreSQL 的聚合函数我们可以在一次查询中选择出符合条件的点并把点聚合成轨迹并找出符合条件的轨迹；如果使用 MongoDB 则要麻烦许多，我们需要做两次查询，第一次查询把所有符合条件的点找出，然后再应用程序中自己写代码把得到的点组织成轨迹，然后第二次查询把轨迹传入数据库进行查询
4. 大数据持久层方案：
 - a) 初步决定采用混合持久化的方案，不同的数据库用来解决不同的问题，

只用一种数据库引擎来应对所有需求较为低效。目前的想法是混合使用关系型数据库（如 PostgreSQL）、NoSQL 数据库（如 Redis、MongoDB、AllegroGraph）、分布式文件系统（如 HDFS）来解决各类大数据的存储

- b) PostgreSQL 等关系型数据库：用来存储结构化和关系型的数据，如土壤养分调查数据；同时 GIS 数据也存储在 PostgreSQL 中，充分利用 PostGIS 的查询功能，如出租车、基站、传感器数据
- c) MongoDB 等文档数据库：插入次数较多的场景，如记录网站访问的日志；存储文档数据，如生产技术数据、病虫害数据
- d) Redis 等 key-value 内存数据库：为关系型数据库中的数据提供分布式缓存服务，将查询的热点数据缓存在内存中，能够大大提高查询速度，同时减少硬盘的读取次数和数据库的压力，如为土壤养分调查数据提供缓存服务；存储会话数据等临时数据，如 VAUD 的连线的中间过程中，用户提交的查询请求后请求的结果以 key-value 的形式缓存在内存数据库中，这样用户在接下来继续连线并查询过程中，可以直接利用中间的计算结果，而免于从头开始查询计算，而一旦会话结束，这些数据就无用了，所以直接从内存中删除，不需要存储到磁盘中
- e) AllegroGraph 等图数据库：用于存储知识图谱等节点-关系结构的数据
- f) HDFS 等分布式文件系统：用于存储图片、视频等大文件数据，由于图片、视频等文件都是顺序的流式访问，使用文件系统存储正好合适；同时，将图片存储在文件系统中，方便前端直接把图片的 url 写入 html 的 img 中，浏览器能够直接显示图片
- g) 直接将数据库封装成服务：以服务的形式提供数据所有权管理及 API，各个应用程序不需要各自同数据库通信，只需要同数据服务通信来查询数据，从而实现应用程序与底层数据库的隔离，应用程序不需要知道底层到底采用了何种数据库，同样的这样无需修改依赖数据的应用程序，即可在服务内部完善数据库，如下图所示



下周计划

1. 搭建数据库服务，连通前端应用程序和后端数据